Database basics

- What, why, how, and what not to do

T.Wildish October 13, 2000

Introduction

- What a federation is.
  - The files and servers that is uses.
- Creating your own 'view' or 'clone' of a federation.
  - Reasons for having your own federation view.
  - Types of views of a federation.
  - How to create your view of a federation.
  - How to share databases among individuals or groups.
- Manipulating federations with Objectivity commands.
  - Summary of some of the more useful Objectivity commands, a bit about locks etc.
  - How to delete your federation - or rather, how *not* to!
- Not an in-depth guide, more a quick tour so you will be familiar with things as you need them or encounter them.
  - Not really needed until you want to go beyond making ntuples or Hbook histograms from 'official' federations.

T.Wildish October 13, 2000

Federation structure - files

- A federation contains several files and directories.
  - A federation file (.FDDB) and a bootfile (.boot).
    - FDDB contains the 'catalogue' of databases, I.e. which databases are in this federation and there physical locations.
    - your starting point for anything is always the bootfile.
  - *Metadata* databases - database files with metadata such as run and event collections.
  - *Data* databases - database files with actual event data.
    - *metadata* and *data* database files are identical from the viewpoint of Objectivity. The distinction is ours alone.
  - A 'journal directory' where 'journal files' are kept.
    - used internally by Objectivity to manage transactions.
- A federation has a 'schema' to describe its contents.
  - If you define your own classes and wish to store them in the federation, you must write a schema.
    - Actually stored in the FDDB, not in a separate file. Can be dumped to a file and loaded into another federation.

T.Wildish October 13, 2000

Federation structure - servers

- Database files are just binary files on a disk that are served by the Objectivity AMS (Advanced Multithreaded Server).
  - One AMS is needed per host that is used to serve any part of the federation.
    - not needed if you run locally, e.g. on a laptop.
  - One AMS is needed to serve the FDDB and bootfile, if they are on a host that is not serving (meta-)data.
- Journal files are also served by the AMS.
  - They are created by the AMS, not by the lockserver.
- Each federation needs exactly one lockserver.
  - Can run anywhere, may be a dedicated host.
  - 'Different' federations can use the same lockserver.
- Never run the servers as root, always as a normal user.
- Federation files, bootfiles, journal files and (meta)data can be served from one host or from many hosts, it's your choice. You should choose your servers carefully.

T.Wildish October 13, 2000

Federation structure - the details

- Federations have a 'federation identifier', a.k.a. FDID.
  - Must be unique for a given lockserver.
  - Federations on different lockservers may have the same FDID.
- Federations have a 'federation name' to identify them too.
  - Used 'internally', need not be unique across federations.
- Federations have a 'pagesize'.
  - Objectivity reads and writes in pages.
  - CMS uses a 32kb pagesize.
- Federation bootfile contains all this information, plus other federation-wide information:
  - Name of lockserver host.
  - Hostname and path of the journal-file directory.
  - Hostname and path of the FDDB file.

T.Wildish October 13, 2000

Database structure - the details

- Databases also have a 'database identifier' (DBID).
  - Unique within one federation.
  - Need not be unique across many federations.
    - even if they share the same lockserver
    - should still be unique if you want to share your DBs
  - Once defined, you should not try to change it.
    - actually maybe you can, but you probably don't want to
- Databases have a 'system name', an identifying string.
  - Also unique within a federation, but not across them.
  - May be changed at any time (not often useful).
- Databases have an 'image path', to specify their location.
  - Contains hostname and full pathname to the physical file.
    - actually stored in the FDDB.
- Databases know about the pagesize of the federation.
  - Cannot attach to a federation with a different pagesize.

T.Wildish October 13, 2000

Viewing the federation catalog(ue)

```
> oodumpcatalog cmsuf01::/cms/reconstruction/user/jetDIGI0300/jetDIGI0300.boot

Objectivity/DB (TM) List Database Files Utility, Version 5.2.1
Copyright (c) Objectivity, Inc 1990, 2000. All rights reserved.

FD Name   = jetDIGI0300
FD ID     = 7000
FD File   = cmsb19::/cms/reconstruction/user/jetDIGI0300/jetDIGI0300.FDDB
Boot File = cmsb19::/cms/reconstruction/user/jetDIGI0300/jetDIGI0300.boot
Jnl Dir   = cmsb19::/cms/reconstruction/user/jetDIGI0300/journal
Lock Host = cmsb19

DB Name   = CARF_System.META.jetmetSignal0300
DB ID     = 11
DB Image  = cmsuf01::/cms/reconstruction/user/jetDIGI0300/\
                CARF_System.META.jetmetSignal0300.jetmetHitPU.DB

DB Name   = LogBook.META.jetmetSignal0300
DB ID     = 12
DB Image  = cmsb19::/cms/reconstruction/user/jetDIGI0300/\
                LogBook.META.jetmetSignal0300.jetmetHitPU.DB
```

T.Wildish October 13, 2000

'Views' of a federation

- I tend to think of a federation as a view of a set of data.
- Many different federations can be constructed providing different views of the same data.
  - 'Production' federation, where MC data is produced.
  - 'User' federation, which users use to access the data.
  - 'Private' federations that users use for a variety of obscure and bizarre purposes (often called 'physics analysis').
- Often these federations will access the exact same physical files to read (most of) the data contained in those federations. The unique parts of each federation will be small. This is why I refer to different federations as merely being different views of the same data.
  - Analogy: 'My YAHOO'. A personal, customised view of what is in fact the same web-site for all users.

T.Wildish October 13, 2000

Types of (private) view of a federation

- Starting from any federation there are several types of private view that you can make:
  - A 'shallow-copy' (à la C++) which points to the original disk locations for all metadata and data.
    - starting point for analysis that needs to create persistent data, tag databases, or eventually deep copies
  - A 'meta-deep' copy, where you copy the metadata locally but still serve event data from the original locations.
    - used mostly in monte-carlo production, to isolate the 'production-view' of a federation from the 'user-view' of the same federation
    - not discussed further in this tutorial
  - A 'deep-copy', where you copy (part of) the event data, and maybe the metadata, to a different physical location.
    - used for WAN-replication of data (copying entire files)
    - can be used to put events on your laptop for later analysis
    - can be used to optimise access to small samples of data

T.Wildish October 13, 2000

Shallow-copy

- Use it when you want to create new databases of your own to ensure that your databases do not pollute the original federation.
- E.g, tag databases.
  - Essentially your own metadata with pointers to events served from their original disk locations (unlike ntuples).
- New persistent objects of your own.
  - E.g. jets created according to your own algorithm.
- Local copy of collections of events (a.k.a. re-clustering).
  - Physically copy (parts of) events to a new database file, i.e. create a deep copy.
- You do not need a shallow-copy if:
  - You only want to read the data that already exists in the federation.
  - You only create transient objects of your own.

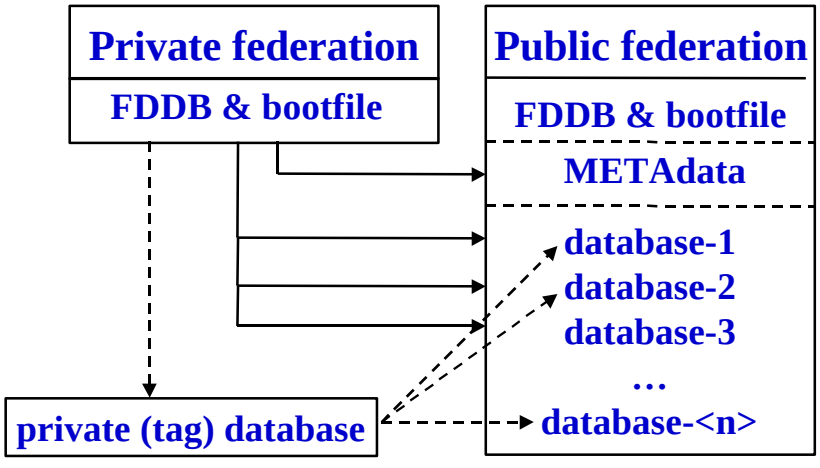
T.Wildish October 13, 2000

Deep-copy

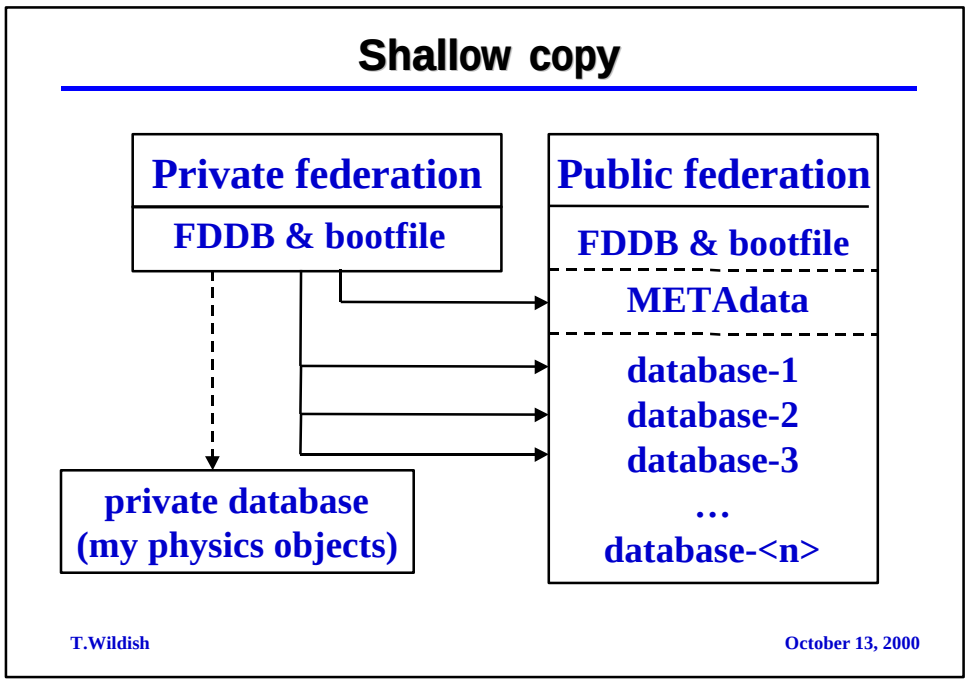
- Deep copies are built on shallow-copies.
  - The shallow-copy is always the starting point.
- Useful later in the cycle of analysis.
  - Once a (sparse) event selection is determined, you can 'localise' (or 'recluster') some or all of the events with a deep copy to shorten the analysis cycle.
    - avoid opening multiple files on multiple servers
    - robust against server downtime
    - efficient use of local disk instead of going to the network
    - avoid competing with other users for staging from MSS
    - can work isolated from the network (e.g. on a laptop)
  - A deep-copy can be whole events or just parts of events (e.g.Digis but not MCInfo), or even entire database files.
    - optimise speed-of-access vs. available local disk etc
- Always retain access to the whole dataset through the original DB specified in the federation.

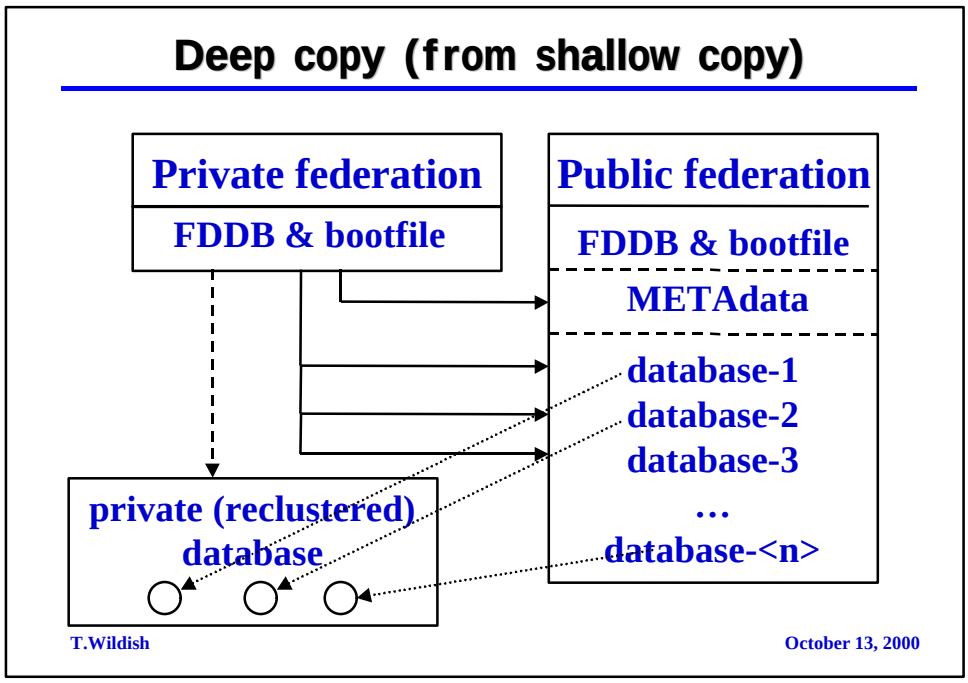
T.Wildish October 13, 2000

Shallow copy



T.Wildish October 13, 2000


Shallow copy
Private federation
FDDB & bootfile
Public federation
FDDB & bootfile
METAdata
database-1
database-2
database-3
…
database-<n>
private database
(my physics objects)
T.Wildish
October 13, 2000


Deep copy (from shallow copy)
Private federation
FDDB & bootfile
Public federation
FDDB & bootfile
METAdata
database-1
database-2
database-3
…
database-<n>
private (reclustered)
database
T.Wildish
October 13, 2000

Creating a shallow copy

- Choose a machine to host the federation.
  - Your own desktop machine will do.
  - You need ~10-20 MB of local disk space, plus eventually space somewhere for the private databases you will add.
    - not on AFS, NFS, or any remote network-server
    - not on scratch space or /tmp either please
  - You need to be running an AMS server on that machine - or someone must be running one for you.
    - The AMS must have write access to the FDDB and the journal directory or you will get 'permission denied' problems
    - use 'oocheckams'/'oocheckls' to check the servers are running
  - You (or someone) must be running a lockserver there, or you must know the name of a lockserver host you can use.
- Copy the FDDB and bootfile of the original federation to the directory that will host your federation, then...

T.Wildish October 13, 2000

Creating a shallow copy (cont'd)

- Use 'ooinstallfd' to make the copied files a usable federation.

```
> ooinstallfd -fdnumber 2907 -lockserverhost pctony -jnldirpath \
/users/wildish/journal jetDIGI0300.boot -nocatalog -notitle -nocheck
```

- You can use 'shallow_federation_copy.pl' in /afs/cern.ch/user/w/wildish/public/bin to help you.

```
> shallow-federation-copy.pl --source \
cmsuf01::/cms/reconstruction/user/jetDIGI0300/jetDIGI0300.boot

Lockserver will be pctony
Using your Unix account ID for the federation ID: 2907
OOInstalling the new federation parameters…
Your bootfile would be pctony::/users/wildish/jetDIGI0300.boot
```

- '--help' will give you the syntax details.
- You can change some of the federation details later with 'oochange'.

T.Wildish October 13, 2000

Creating a deep copy

- See the tutorial examples (Stephan Wynhoff) and/or the ORCA Users Guide.

T.Wildish October 13, 2000

Sharing databases between federations

- Problem: I have my own database of persistent objects (e.g. 'myjets') attached to my shallow copy, and Joe User has 'hisjets'. We want to compare them in the same job. This means we need to merge them into the same federation.
  - Basically I need a partial deep-copy, but of part of another private federation instead of a public one.
  - No mature, user-friendly tool for this, yet.
- Big problem: DBID-uniqueness.
  - If Joe & I work in identical shallow copies of the same federation, our new databases will have the same DBIDs.
    - shallow copies don't know about databases in the other federation, so new databases get the next available DBID
  - Since DBIDs are unique in a federation, I can't put my and Joe Users' new databases into the same federation.
    - not 100% true. Can change DBID providing no data in the federation knows about the contents of this database. E.g. no metadata pointing to this database. Best avoided!

T.Wildish October 13, 2000

Sharing databases... (cont'd)

- Solution: pre-allocate a range of DBIDs in one shallow copy before creating private databases, leaving that range free for other users.
- Basic idea: run 'oonewdb'/'oodeletedb' on one federation repeatedly to pre-allocate or 'use up' a set of DBIDs, leaving them free for use in other federations.
  - Tools do exist to automate this.
    - but you should think to do this before you start creating your own databases!
    - not mature or user-friendly yet (coming soon).
    - if you need to do this, contact me first for help.
- Objectivity 6.0 will make this easier to deal with.
  - Provides an API to allow you to choose your DBID.
    - not yet available, currently in beta (October 2000).
    - must still be aware of, and manage, the DBID problem. However they are chosen, they will be unique in a federation.

T.Wildish October 13, 2000

Deleting your private federation

- The command 'oodeletefd' exists to delete federations.
- NEVER use it.
- Not even on private federations.
  - 'oodeletefd' will delete the federation file, the bootfile, and *all* the database files. Since any private federation will still get many of it's files from the original locations, this is Bad News.
  - Objectivity has a 'security-API' to address this type of problem, we will deploy it as soon as we can.
- You can delete federations quite happily with 'rm -rf'.
  - The AMS may cache data still for a while, but it will eventually forget. A restart of the AMS will ensure this.
  - Should use 'oolockmon' and 'oocleanup' first to remove stale locks.
- Not to be confused with 'oodeletedb', which is fine for use on private federations.

T.Wildish October 13, 2000

Locks...

- When you open a federation for read or write access, Objectivity takes out locks on the federation and the database files you access.
  - Ensures that data is not modified while you are reading it.
  - Ensures that two people do not modify a database at the same time.
- Objectivity creates and deletes locks automatically, you never need to worry about them…
- …unless something goes wrong.
  - If your job crashes badly, Objectivity may not be able to clean up the locks it has created.
  - Further access to the same databases may be blocked because of these 'stale' locks.

T.Wildish October 13, 2000

Viewing locks

- You need to know how to detect and remove stale locks.
- Use 'oolockmon <bootfile>' to view the locks.
- For each lock you get:
  - The PID of the process holding the lock.
  - The name of the host that it is running on.
  - The UID of the user running the process.
  - The 'transaction ID' corresponding to the lock.
  - The type of lock (read, update etc)
  - If the lockserver is serving many federations, it will report about locks in all of them.
  - …and more...

T.Wildish October 13, 2000

Finding stale locks

```
> oolockmon jetDIGI0300.boot
Lock Table for Lock Server on node "pctony" - your UID is 2907

    UID     HostID   PID    TransID  APID  Mode  Type FdbID  DbID OclID
  ------------------------------------------------------------------------
   2907     pctony 29842    1310719 65535  read   Ocl  2907     1     4
   2907     pctony 29842    1310719 65535  read   Ocl  2907     1     3

Lock table displayed, 2 entries.
```

- Look for locks held by your UID against your federation.
  - The bootfile or oodumpcatalog will tell you the FDID.
- Determine if the lock is stale or not.
  - If the given PID is still running on the given host, the lock is not stale.
    - do not 'oocleanup' such locks!
  - If the process is no longer there, or is a zombie, you must cleanup the lock using 'oocleanup'.

T.Wildish October 13, 2000

Removing stale locks with 'oocleanup'

- Can clean up individual locks using the transaction ID.

```
> oocleanup -transaction 1310719 jetDIGI0300.boot
Recovered transaction 1310719 for FD "jetDIGI0300.boot".

Finished processing all autonomous partitions.
Finished attempting to recover all specified transactions.
```

- Can clean up all the locks on any host using the '-local' flag.
  - Useful to run at the end of a batch job in productions (if 200 batch jobs die simultaneously, you have a mess!).
  - Not so useful to run at the end of other types of job.

```
> oocleanup -local jetDIGI0300.boot
Recovering local transactions for FD "jetDIGI0300.boot" on host "pctony".
** Error #3104: oocleanup: Failed to recover transaction 1507327 for FD
                "jetDIGI0300.boot". Error #0: .

Finished processing all autonomous partitions.
Finished attempting to recover all specified transactions.
```

T.Wildish October 13, 2000

Removing stale locks (cont'd)

- 'oocleanup' has many other flags.
  - '-force', '-deadowner' etc.
- Don't use these unless you have to.
  - I.e. don't use them unless cleaning by transaction ID or with '-local' fails.
- Don't use them until you have read the help or the manual ('oocleanup -help').

T.Wildish October 13, 2000

Environment variables for Objectivity

- There are many environment variables you can use to control details of Objectivity operation.
  - You should not need to care about any except OO_FD_BOOT and OO_RPC_TIMEOUT.
- OO_FD_BOOT sets the 'default' bootfile for operations such as oodumpcatalog, oochangedb etc.
  - This can be highly dangerous!
    - if you use, e.g, oodeletedb without giving the bootfile, and OO_FD_BOOT is not what you think it is, you may damage another federation by mistake.
  - I recommend you only use OO_FD_BOOT to point to your own federations, never to a federation controlled by someone else.
    - for interactive sessions that is. I hope you never run oodeletedb in batch!

T.Wildish October 13, 2000

Environment variables (cont'd)

- OO_RPC_TIMEOUT controls the amount of time that a task will wait for a response from the AMS.
  - Default value is 10 (units are seconds), which is good enough in most cases.
- If you get 'timeout waiting for…' errors, set it a little higher.
  - Should never need to set it above ~120.
- This is not the whole story.
  - AMS can 'defer' a client for much longer if it knows it will take time to retrieve the data (e.g. from MSS).

T.Wildish October 13, 2000

Useful Objectivity commands

- oodumpcatalog
  - List the contents of a federation.
- ooinstallfd
  - Create a new private shallow copy of a federation from an FDDB and bootfile copied from an existing federation.
- oodeletefd - NEVER USE THIS!
- ooattachdb (only on your own federations)
  - Attach a database file to a federation (after creating it in another federation).
- oochangedb (only on your own federations)
  - Change the DB System name and or filepath (and host) for a database that already exists in the federation.

T.Wildish October 13, 2000

Useful Objectivity commands (cont'd)

- oocleanup (any federation, but with care)
  - Clean up locks left if (when) jobs crash.
- oolockmon
  - Find out what locks exists on a given federation.
- oonewdb (only on your own federations)
  - create a new database with a given name. Not normally needed
- oodeletedb (only on your own federations)
  - Remove databases from a federation.
    - can be used with oonewdb to pre-allocate DBID ranges
- ooschemadump
  - Dump the schema from a federation to a file.
- ooschemaupgrade (only on your own federations)
  - Upgrade the schema of a federation to know about new types of persistent objects.

T.Wildish October 13, 2000

Summary: creating private federations

- Create a shallow copy of a federation when you want to:
  - Create your own tag databases to narrow down the event selection.
  - Create your own persistent objects of classes you have defined.
  - Create a deep-copy. I.e. copy a (small) sample of the data to a local disk to improve performance accessing them (re-cluster them).
- Private federations can be considered as playgrounds.
  - Create them, play with them, debug your private tags or objects, delete them.
- Think carefully about who you want to share with!
  - The DBID-uniqueness requirement means you will always have to think who will use the data you create.
- Think carefully before changing your federation 'by hand'.
  - Especially w.r.t. using OO_FD_BOOT by default.

T.Wildish October 13, 2000